Integration of New Proprietary Symmetric Key Encryption Algorithm with MS Outlook Email Client

A Thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of
Master of Technology

by

M A Jaffer Ali
Roll No. : 14111051

under the guidance of

Prof. Manindra Agarwal

Department of Computer Science and Engineering
Indian Institute of Technology Kanpur

July, 2016

CERTIFICATE

It is certified that the work contained in the thesis entitled **Integration of New Proprietary Symmetric Key Encryption Algorithm with MS Outlook Email Client**, by **M A Jaffer Ali**, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Prof. Manindra Agrawal
Department of Computer Science & Engineering
Indian Institute of Technology
Kanpur – 208016.

Abstract

Email being most used and regarded network services, has become the primary means of communication in many organisations and commercial entities. Hence the need for the confidentiality of the information exchanged via email has increased. Presently, standard cryptographic techniques and formats like Pretty Good Privacy(PGP) and Secure/Multipurpose Internet Mail Extensions(S/MIME) are used to protect the e-mail messages.

In this thesis, a new proprietary symmetric key encryption algorithm is used to encrypt the email messages originated via MS outlook 2010 email client. Both symmetric key encryption and asymmetric key encryption algorithm is used to ensure confidentiality of the message. The message is encrypted using a symmetric key encryption algorithm, which needs a symmetric key. The symmetric Key is also called as session key and each symmetric key is used only once. The encrypted message and its session key are sent to the recipients of the email message. The session key must be sent to the email recipients so that, they know how to decrypt the message. The public key encryption algorithm is used to securely share the session key with the recipients of the email. The session key is encrypted with the recipient's public key which can be decrypted only with the private key belonging to the recipient. All the necessary APIs' for integration of the MS Outlook with the new proprietary symmetric key encryption algorithm is identified and used for developing the application that ensures confidentiality of the email message.

Acknowledgments

I wish to express my profound gratitude to my thesis advisor, Prof. Manindra Agarwal, for continuous guidance and support provided during my thesis work. Despite his busy schedule, he was always available for discussions. He gave total freedom in executing my work. His guidance wasn't confined to academic issues and his advice has always been enlightening.

I would like to thank all officers of Indian Air Force who worked hard to get me an opportunity to pursue M. Tech here at IIT Kanpur.

I would also like to thank all my course mates especially K Ganesh Kannan, who gave many constructive suggestions throughout the course of my work and my lab mate Shivmaran who helped me to clear the issues with the code. Finally, I would like to thank my family, which has always been my greatest support and inspiration.

M A Jaffer Ali

July, 2016

Dedicated to

My family who are my greatest blessing

He who would learn to fly one day must first
learn to stand and walk and run and climb and dance;
one cannot fly into flying.

– Friedrich Nietzsche

Contents

Chapter 1

Introduction

1.1 Overview

For the last few decades, one has become increasingly reliant on email. Unlike real mail, it is ubiquitous and swift. E-mail application is the most widely used and regarded network application. It is significant, because it is used in business, defence, health and educational institutions etc for exchange of critical information such as business information, weapon details, health patient record and so on.

The early e-mail systems were very simple and straightforward. It simply offered the basic e-mail functions without any security services. But today,many people and organizations started using e-mail for exchange of information and few organisation even use it for exchange of sensitive information. As a result, the e-mail systems became the target to many threats and eavesdroppers. Hence, the need for highly-secured e-mail systems increased remarkably. Thus it is necessary to reevaluate the properties of e-mail security and enhance the security services that apply those properties in the e-mail systems. These enhancements[Sta99] include the following:

1. **Confidentiality:** This property assures that, the transmitted information can be accessed only by the authorised party. It also protects the transmitted data from passive attacks and traffic flow analysis.

2. **Integrity:** This property assures that, the transmitted information can be modified only by the authorised party. The property also covers the destruction of data.

3. **Authentication:** This property assures that the origin of the email is correctly identified and with an guarantee that the identity is right.

4. **Non Repudiation:** This property preclude either sender or recipient from denying the transmitted message. Thus when the message is sent, the recipient can prove that the message was infact sent by the alleged sender. Similarly, when a message is received, the sender can prove that the message was infact received by the alleged recipient.

Currently standard cryptographic techniques and formats like PGP and S/MIME[Sta99] are used to protect the email messages.

1.2 Contribution of the thesis

In this thesis, an approach similar to PGP is followed wherein a new proprietary symmetric key encryption algorithm is used to encrypt the email messages originated via MS outlook 2010 email client. Both symmetric key encryption and asymmetric key encryption algorithm is used to ensure confidentiality of the message. The email is encrypted using a symmetric key encryption algorithm, which needs a symmetric key. The symmetric key is also called as session key and each session key is used only once. The encrypted message and its session key are sent

to the recipient of the email message. The session key must be sent to the recipients so that they know how to decrypt the message. The public key encryption algorithm is used to exchange the session key with the recipients of the email. The session key is encrypted with the recipient's public key which can be decrypted only with the private key belonging to the recipient. All the necessary APIs' for integration of the MS Outlook with the new Symmetric Key encryption algorithm is identified and used for developing the application. The Visual Studio 2015 is used as an IDE for development of the application and the programming has been done in Visual CSharp. The necessary modifications in the user interface of the outlook application has been carried out.

1.3 Organisation of the report

In chapter two, DotNET Framework Classes and Methods used to perform the various operations are described and in chapter three the process adopted to Encrypt/Decrypt the email message has been discussed. Finally in chapter four we conclude by summarizing our work and stating the possible future directions this thesis seems to suggest.

Chapter 2

Dot Net Framework Classes and Methods

The DotNET Framework is a managed execution environment, which provides different services to its running application. Common Language Runtime of this framework provides memory management and system services. It has an extensive class library, which makes the job of application developers far more easy. In this chapter, we introduce the classes and methods that are used to integrate the new proprietary symmetric key encryption algorithm for encryption/decryption of the email messages originated via MS Outlook client 2010.

2.1 Running the New Encryption algorithm

The proprietary algorithm is available in the form of executable. To run the .exe of Encryption algorithm, the following properties/method of Process class[MSDa] and ProcessStartInfo[MSDb] class of System.Diagnostics namespace is used. The encrypted message returned by the algorithm is appended as an attachment to

the mail item. The same properties/method is also used to decrypt the encrypted message.

- **Process Start Info:** This constructor specifies a set of values that are used when you start a process.
- **ProcessStartInfo(String):** This constructor initializes the new instance of process start info class and states the file name with which to start the process.
- **UseShellExecute:** Use Shell Execute property of the Process start Info class indicates whether to use the operating system shell to start the process. This property is set to false as the process should be created directly from the executable file.
- **RedirectStandardOutput:** Redirect standard output property of the Process StartInfo class indicates whether the textual output of an application is written to the Process.standard output stream. When a Process writes text to its standard stream, that text is typically displayed on the console. By setting RedirectStandardOutput to true to redirect the StandardOutput stream, you can manipulate or suppress the output of a process. For example, you can filter the text, format it differently, or write the output to both the console and/or to a designated log file. The Use Shell Execute property has to be set to false if we want to set Redirect Standard Output property to true.
- **CreateNoWindow:** This property of ProcessStartInfo class indicates whether to start the process in a new window. This property is set to true, if the process should be started without creating a new window to contain it.
- **WorkingDirectory:** When the UseShellExecute property is false, this property specifies the working directory for the process to be started.

- **Arguments:** This property specifies the set of command-line arguments to use when starting the application. Arguments are parsed and interpreted by the target application, so must align with the expectations of that application. For DotNET applications spaces are interpreted as a separator between multiple arguments. A single argument that includes spaces must be surrounded by quotation marks, but those quotation marks are not carried through to the target application.

- **Start Method:** Start method of a Process class starts a process that is mentioned by the StartInfo property.

2.2 Connecting to SQL Database Server

SQL express server is installed to store the public key of the email users. A table as shown below, is created to store the email Id (Primary Key) and the Public Key of every user in the database server.

Email ID	Public Key
jaffermce@gmail.com	RSA Public Key
jafarmce@gmail.com	RSA Public Key

TABLE 2.1: Table to store Public Key

The first thing to do, to connect to the database server is to create a connection. The connection then specifies to the the rest of the ADO.NET code about the database that it is talking to. ADO.NET[May] is the new database technology of the DotNET platform which is build on Microsoft ActiveX Data Objects (ADO). ADO.NET is an integral part of the DotNET Framework and provide access to relational data, application data and XML documents. ADO.NET provides data access services in the Microsoft DotNET platform. It supports different types of development needs.

DataSet and DataTable objects are defined in ADO.NET. These objects are optimized for moving disconnected sets of data across Internets and Intranets, including through firewalls. It also includes the traditional Connection and Command objects, as well as an object called a DataReader which is used to fetch the data from the database server.

One can use ADO.NET to access data by using the new DotNET Framework data providers. System.Data.SQLClient is the Dot NET framework data provider for the SQL server. ADO.NET contain classes that expose data access services to the DotNET developer.

There are two important components of ADO.NET classes:

- DataSet
- Data provider

Data Provider is a set of components that includes Connection object(SQL Connection), Command Object (SQL command), Data Reader Object (SQL Data Reader) and the Data Adapter Object (SQL Data Adapter). These objects are discussed below:

- **SQL Connection Class:** This class handles connection to the database server. The SQLConnection constructor instantiates this class. The connection string specifying the data source and database name can also be passed to this constructor.

  This object signifies a unique session to the SQL Database server. SQL Command and SQL Datareader objects are used together with SQL connection object to enhance the performance when connecting to the database server.

- **SQL Command Class:** This class is used to execute the SQL command on the database server. This class sends the command to the database

server that is specified by the connection string of connection class. The SQLCommand constructor is used to instantiate this class. One can Query, modify, delete, insert and update the data in the Database server using this object.

- **SQL Data Reader Class:** This class is used to read database records one-by-one in forward order from an SQL database. To read the data using SQL data reader objet, first a connection is established with the SQL server. Then using SQLCommand object call its ExecuteReader method, assigning the reference it returns to an SQLDataReader.

  The SQLDataReader is fast because of its forward only design. While traversing the data or writing the data back to the data source it doesn't carry any overhead. If the data is to be read only once then SQLDataReader is the fastest method available. Streaming behaviour of the class is used when the data need to read is larger than the data that is preferred to be stored in the memory.

  The data is returned by SQLDataReader in a sequential stream. To read the data, it must be pulled from the table, record by record. The details of the previous record is not maintained once the new record is read. If the previous record has to be read then the new instance of SQL reader is to be created.

  While loop as shown below is used to iterate through the each row of data stream that the SQLDatareader returns,to read all the data.

```
while (rdr.Read())
{
        // get the value of each field
                string pk = Convert.ToString(rdr["publicKey"]);

        //subsequent code
}
```

In the above code, in the while loop, Read method is called on the SQL-DataReader, rdr. The Read method returns true when there are more records to read. Read method returns false once all the records are read.

2.3 Modifications carried out in Outlook

The following modifications are carried out in Outlook user interface.

- To the Compose window ribbon of the Outlook, a tab named "Security" is added. A ribbon is way of organising the program features in to series of tabs. A ribbon usually appears at the top of the window. A ribbon allows to find/locate the function and feature button easily. Several function buttons are added to every tab. The button named "Encrypt" is added to the tab. The click of this button starts the encryption process.

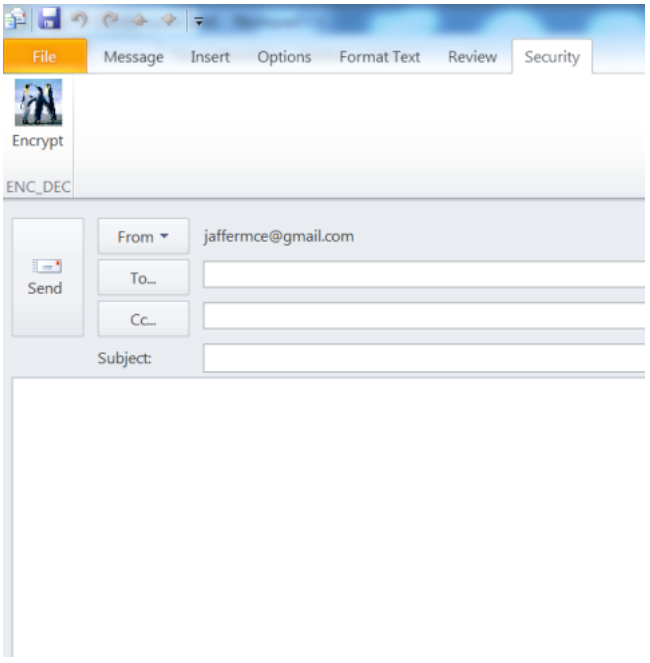

FIGURE 2.1: Compose window modification

- To the Read window ribbon of the Outlook, a tab named "Security" is added. The button named "Decrypt" is added to the tab. The click of this button starts the decryption process.

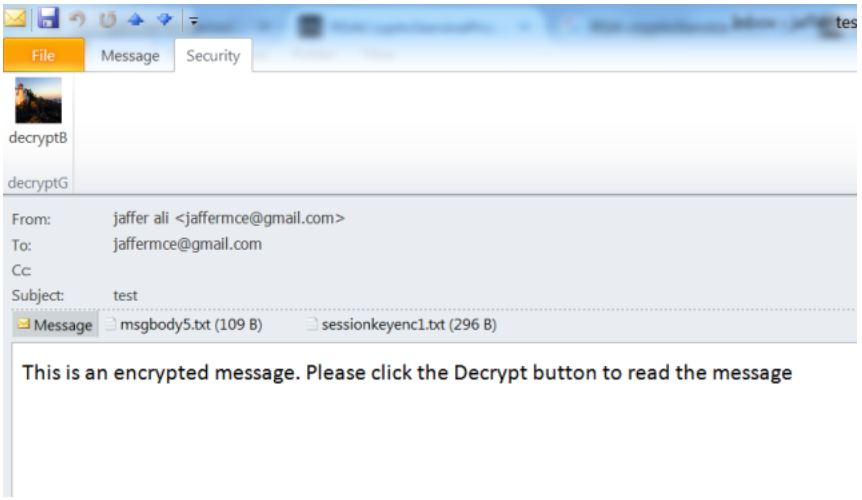

FIGURE 2.2: Read window modification

2.4 Generation of Session Key

For encrypting the email message including attachments with the new symmetric key encryption algorithm, a session key of 256 bits length is generated. The DotNet Framework provides a Random class[Dot] in System Namespace to generate the session key.

Random class represents a pseudo-random number generator. It is a device that produces a sequence of numbers that meet certain statistical requirements for randomness. These numbers are selected with equal probability from a finite set of numbers. The selected numbers are not fully random because a mathematical algorithm is used to select them, but for all practical purposes they are sufficiently random. The present implementation is based on a modified version of Donald E. Knuth's subtractive random number generator algorithm.

The seed vale is provide to random number generator while instantiating a Random class constructor. The seed vale act as a starting value for the algorithm. The seed value can be supplied either explicitly or implicitly.

- The Random(Int32) constructor uses a seed value that is supplied explicitly.
- The Random() constructor uses the system clock value as a seed value. This way is mostly used.

Random.Next(Int32) method returns a non-negative random integer. The value is less than the value that is passed as argument to this method. We make repeated calls to the Next method to generate a session key of requisite length. The Session Key that is generated is a string of 32 characters. The 32 characters are randomly selected from a restricted ASCII code set. The code set is restricted to avoid the encoding/decoding issue.

2.5 Generation of Asymmetric Key pair

RSACryptoServiceProvider[MSDc] class of DotNET Framework is used to generate the asymmetric key pair. On instantiating the default constructor of this class it automatically create an asymmetric key pair. The key pair is stored for multiple sessions or for a period as specified by the security policy. The following methods are use to extrat the key thus generated.

- **ToXmlString method**, returns an XML representation of the key information.
- The **ExportParameters method**, returns an RSAParameters structure that holds the key information.

When boolean value true is passed to these methods it returns both public and private key information. When false is passed to these methods, it returns only the public key information. The private keys should never be stored on the local computer. Key container should be used to store the private key.

2.6 Encryption/Decryption of Session Key

The session key generated for encryption of the message is further encrypted using RSA algorithm for the purpose of transmission of session key to the recipient of the email message. The DotNET Framework implements the RSA algorithm in the **RSACryptoServiceProvider** class. The instance of this class lets you create Key pair(as discussed in the Generation of Asymmetric Key section), encrypt using a public key and decrypt using a private key.

A method named **RSAEncrypt**[MSDc] is defined for the purpose of encryption. The session key and the recipient public key are passed as an input arguments to the method. A RSA Crypto Service provider class is instantiated inside the method. The public key of the recipient fetched from the SQL server is assigned to the RSA object using the **FromXmlString** method. The FromXmlString method initializes an RSA object using key information in an XML string that was generated using the ToXmlString method. The FromXmlString method accepts either an XML string containing a public key or an XML string containing a public and private key. The **RSA Crypto Service provider Encrypt** method is used to the encrypt the generated session key with the public key of the recipient. Before returning the encrypted session key, the dispose method is used to releases all resources used by the current instance of the Asymmetric algorithm class.

To decrypt the session key a method named **RSADecrypt** [MSDc] is defined. The encrypted session key and the private key of the recipient are passed as an input parameter to the method. All steps similar to RSAEncrypt method is followed except for RSA Crypto Service provider Encrypt method, the **RSA Crypto Service provider Decrypt method** is used. The decrypted session key is returned back.

2.7 Cipher Modes

The new symmetric key encryption algorithm provides the following block cipher modes of operation.

- **Electronic Code Book Mode(ECB)**: In this mode the given plaintext is is divided in to blocks of 64 bits and each block of plain text is encrypted using the same session key. The last block is padded is necessary. For a given session key, this mode produces the unique cipher text for every 64-bit block of plain text.Hence it is called codebook. For Decryption one block is taken at a time. Decryption is performed using the same key. The shortcoming of this mode is that the same 64 bit of plain text produces the same cipher text.

- **Cipher Block Chaining Mode (CBC)**: This mode overcomes the shortcomings of the ECB. This mode produces different ciphertext blocks even if the same plaintext block if repeated. This is achieved by XOR-ing the current plaintext block with preceding ciphertext block. The key used is the same for every block. While decrypting the plaintext is obtained by passing each cipher block through a decryption algorithm and XOR the output with the preceding cipher text block.

The mode needs to be set based on the requirement of the application that uses the algorithm. The Electronic Code Book (ECB) mode has been set for email security.

Chapter 3

Encryption and Decryption Methodology

3.1 Introduction

In this thesis, the new proprietary symmetric key encryption algorithm is integrated with the MS outlook email client for ensuring the confidentiality of the email messages. The need for new symmetric key encryption algorithm is, to use the indigenously developed algorithm for encrypting/decrypting the mission critical information of highly sensitive organizations. MS outlook is selected as a preferred email client for this task, because of its wide use and rich availability of development tools. Visual Studio 2015 is used as the integrated development environment and the programming is done in Visual C Sharp.

3.2 Encryption Process

The email message originated via MS outlook email client is encrypted with the new symmetric key encryption algorithm on click of the Encrypt button under

Security tab of the compose window. The user on opening the compose window fills the 'To' field, Subject and then adds the message in the message body followed by attachments, if any.

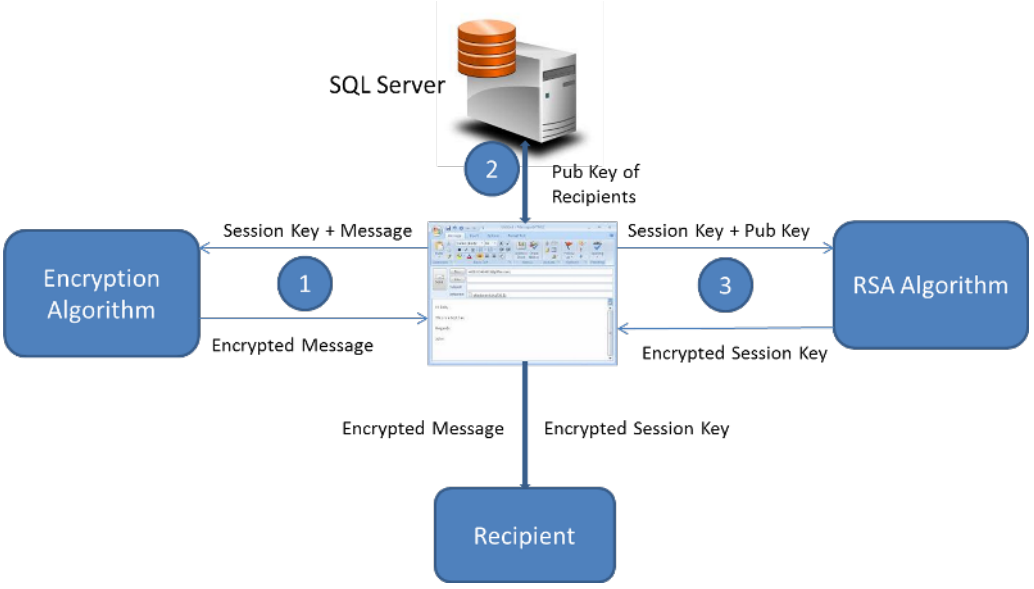

FIGURE 3.1: Encryption Process

If the message needs to be encrypted, the user has to click on the encrypt button under the Security tab. On click of the encrypt button the following process begins:

1. A 256 bit length session key is generated.

2. The message body of the compose window is copied to a file, named "msgbody.txt" and file is appended as one more attachment to the email. On completing the above action, "msgbody.txt" is deleted from the local computer. The message body is replaced with the following text.

   "The Message is encrypted. Please click on the decrypt button to read the message".

3. The attachments and the generated session key are passed as an input to the new symmetric key encryption algorithm. For this, first all the attachments are copied to a local directory, created for this purpose. The following arguments needs to be passed to the new algorithm:

   - The session key
   - The input file (the message that needs to be encrypted/decrypted)
   - The location of output file, where the encrypted/decrypted data needs to be saved
   - E/D(E for encrypt and D for decrypt)
   - Electronic Codebook/Cipher Block Chaining(ECB/CBC cipher modes)

   The encrypted attachments (including message body) are stored at a specified location. All these encrypted files are appended as an attachment to the mail item. The original attachments (in plain text) are deleted from the mail item so that the compose window will only contain the encrypted attachments and encrypted message body as an attachment.

4. The public key of all the recipients (from 'To' filed) is fetched from the SQL Express server one after another.

5. The generated Session key and Public key of the each user is passed as an input to the RSA encryption algorithm. The email address of the user is appended to the output of the encryption algorithm (in the format as shown below) and written to a text file. This process is repeated for all the recipients of the email message. The same text file is appended with the email address and encrypted session key for every user.

6. The file containing the encrypted the session keys are appended as a final attachment to the mail item (i.e. compose window). Now the encrypted message is ready to be dispatched.

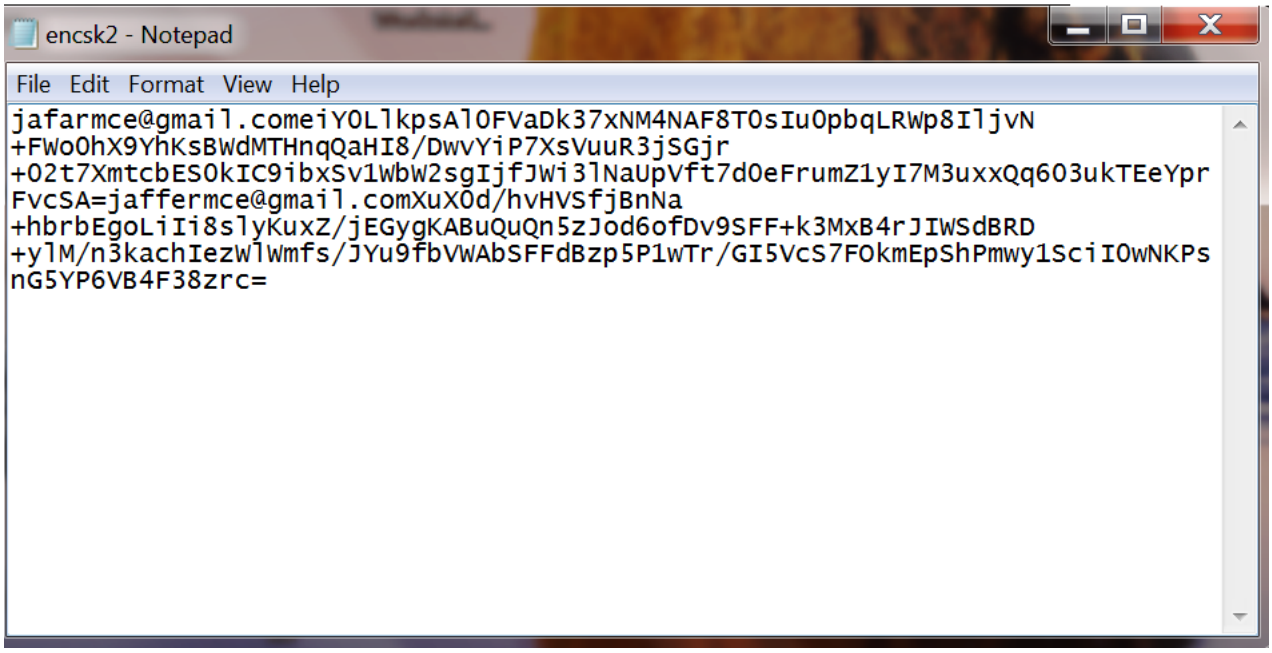

FIGURE 3.2: Encrypted Session Key File

7. Final thing to do is to clean all the files that are created in the process of encryption. This job is linked to the Send click event. When the users click on the Send button all the files that are created to in the process are deleted from the local computer.

3.3 Decryption Process

The message body of the encrypted message will contain the following text.

"This message is encrypted. Please click on the decrypt button to read the message".

On trying to open the attachment before decryption, the user will see, only the junk information. The encrypted message will contain the encrypted attachments, encrypted message body, encrypted session key as an attachment to the mail item. To decrypt the encrypted message, the Decrypt button under Security tab of Read

window is to be clicked. On click of the decrypt button the following process begins:

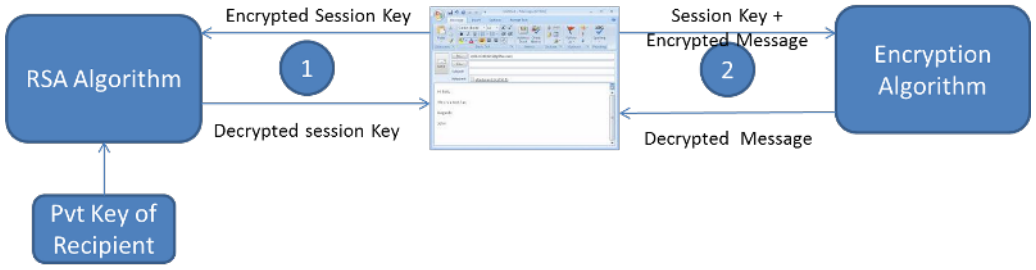

FIGURE 3.3: Decryption Process

1. All the attachment to the mail item is stored at a specified location in the local computer.

2. The availability of the file containing encrypted session key is checked. If the file is not found, then the following error message is displayed.

   "The message cannot be decrypted without Session key file"

   If the file is found, then the encrypted session key corresponding to the recipient needs to be traced. This is done with the help of email address of the recipient. The email address of the recipient is searched in the file and the index of the email address is found. The fixed length string following the email address of the recipient is stored to a variable. This variable now contains the encrypted session key of the recipient.

3. The encrypted session key and the private key of the recipient are passed as an input to the RSA decryption algorithm. The user browses to the file location of the private key. The output of the decryption algorithm will be the 256 bit length session key. Thus the session key is retrieved. The file containing the encrypted session key is deleted from the local computer.

4. The session key, encrypted attachment, the location for decrypted file, D (for decrypt) and cipher mode (ECB/CBC) are passed as input arguments to the

new symmetric key algorithm. The output (decrypted attachment) is stored at the specified location in the local computer. The process is repeated for all the attachments one after another.

5. The message body is replaced with the decrypted message body content.

6. The decrypted attachments are appended to the mail item (inclusive of message body attachment). The original encrypted attachments are deleted from the read window.

7. Finally the files created in the process of decryption are deleted from the local computer.

The code for encryption and decryption process has been written in C Sharp language. For encryption process approximately three hundred lines of code is written and for decryption process approximately two hundred lines of code is written.

Chapter 4

Conclusions and Future Directions

4.1 Summary/Conclusion

The Confidentiality of the email message is ensured by using both Symmetric and Asymmetric encryption algorithm. The benefit of this approach is as described below.

- We are encrypting the email using the symmetric key algorithm and share the symmetric (session) key with the recipients of the email using asymmetric encryption algorithm. This is because the asymmetric encryption algorithms are quite computationally intensive and some algorithms can produce cipher text up to twice the size of the original text and therefore it is not used for encrypting the email message.

- Session Key used for encryption of the mail, sometimes referred to as a nonce (number used once) is completely random and essentially 'unguessable'. Having a good source of randomness (a.k.a. entropy) is very important in system security.

Thus email message is encrypted symmetrically with the session key, and then the session key is encrypted with public key of the recipients. The session key will probably be much shorter than the message itself, so the computational load of asymmetric encryption is lightened significantly. The encrypted session key is attached to the mail item, and the session key is deleted forever. Upon receiving the encrypted document, the private key is used to decrypt the session key. Then use this session key to decrypt the entire message.

4.2 Future Directions

Based on our work, we propose the following future directions for effective deployment of the application developed.

- Embed the private key in the specialised hardware and integrate with the application developed.
- Address the encoding/decoding issue.
- Include other aspects of email security like message integrity, non-repudiation and authentication.

Bibliography

[Dot] DotNetPerls. Random class. http://www.dotnetperls.com/random.

[May] Joe Mayo. Introduction to ado.net. http://www.csharp-station.com/Tutorial/AdoDotNet/Lesson01.

[MSDa] MSDN. Process class. https://msdn.microsoft.com/en-us/library/system.diagnostics(v=vs.110).aspx.

[MSDb] MSDN. Processstartinfo class. https://msdn.microsoft.com/en-us/library/bfbyhds5(v=vs.110).aspx.

[MSDc] MSDN. Rsacryptoserviceprovider class. https://msdn.microsoft.com/en-us/library/system.security.cryptography.rsacryptoserviceprovider(v=vs.110).aspx.

[Sta99] William Stallings. *Crytptography and Network Security*. Prentice Hall International,Inc., 1999.